

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-063508

(43)Date of publication of application : 06.03.1998

(51)Int.Cl.

G06F 9/45

(21)Application number : 08-237211

(71)Applicant : NEC CORP

(22)Date of filing : 19.08.1996

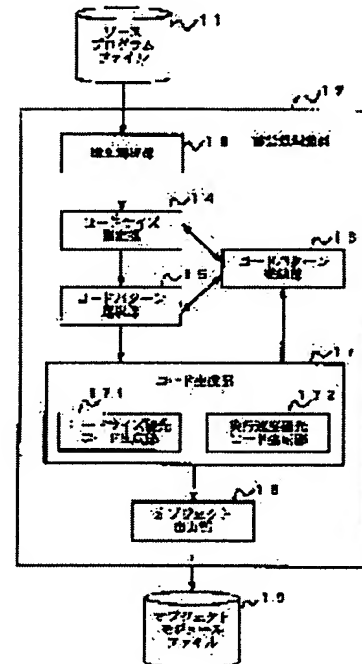
(72)Inventor : EGASHIRA AKIRA

## (54) LANGUAGE PROCESSOR AND ITS METHOD

## (57)Abstract:

PROBLEM TO BE SOLVED: To provide a language processor and its processing method generating the object of a high executing speed within the limit of a prescribed code size.

SOLUTION: The language processor with a syntax analyzing part 13, a code generating part 17 and an object output part 18 is provided with a code pattern registering part 16 for registering prescribed information concerning a pattern outputting different codes depending on a generation code giving priority to the down-sizing of an object and a generation code giving priority to the acceleration of the execution of the object, a code size measuring part 14 measuring the size of the object and the number of the times of using each pattern of outputting codes registered in the part 16, and a code pattern selecting part 15 replacing a part and all of the generation codes giving priority to the execution speed of the object with the generation codes giving priority to the size of the object to be not larger than a prescribed code size.



## LEGAL STATUS

[Date of request for examination] 19.08.1996

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3237693

[Date of registration] 05.10.2001

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-63508

(43) 公開日 平成10年(1998) 3月6日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/45

識別記号

庁内整理番号

F I

G 0 6 F 9/44

技術表示箇所

3 2 2 F

審査請求 有 請求項の数 6 F D (全 13 頁)

(21) 出願番号 特願平8-237211

(22) 出願日 平成8年(1996) 8月19日

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 江頭 朗

東京都港区芝五丁目7番1号 日本電気株式会社内

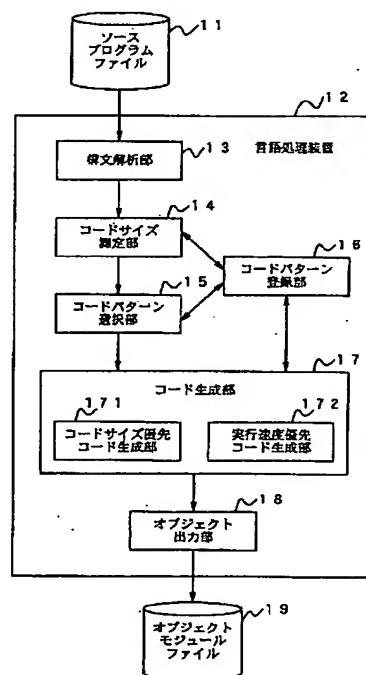
(74) 代理人 弁理士 松本 正夫

(54) 【発明の名称】 言語処理装置および言語処理方法

(57) 【要約】

【課題】 所定のコードサイズの範囲内で実行速度の速いオブジェクトを生成することができる言語処理装置および言語処理方法を提供する。

【解決手段】 構文解析部13と、コード生成部17と、オブジェクト出力部18とを備える言語処理装置において、オブジェクトのサイズを小さくすることを優先した生成コードとオブジェクトの実行速度を速くすることを優先した生成コードとで異なるコードを出力するパターンについて所定の情報を登録するコードパターン登録部16と、オブジェクトのサイズとコードパターン登録部16に登録された出力コードの各パターンを使用した回数とを測定するコードサイズ測定部14と、所定のコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部15とを備える。



## 【特許請求の範囲】

【請求項1】 ソースプログラムを入力して解析する構文解析部と、解析された結果に基づいて命令コードを生成するコード生成部と、生成された命令コードをオブジェクト・モジュール・ファイルに出力するオブジェクト出力部とを備える言語処理装置において、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておくコードパターン登録部と、

オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、前記コードパターン登録部に登録されている出力コードの各パターンを使用した回数を測定するコードサイズ測定部と、前記コードサイズ測定部の測定結果と前記コードパターン登録部の登録内容を参照し、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部とを備えることを特徴とする言語処理装置。

【請求項2】 前記コードパターン登録部が、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録することを特徴とする請求項1に記載の言語処理装置。

【請求項3】 前記コードパターン選択部が、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定することを特徴とする請求項1または請求項2に記載の言語処理装置。

【請求項4】 ソースプログラムを入力して解析する第1のステップと、解析された結果に基づいて命令コードを生成する第2のステップと、生成された命令コードをオブジェクト・モジュール・ファイルに出力する第3のステップとを有する言語処理方法において、前記ソースプログラムを解析する第1のステップと命令コードを生成する第2のステップとの間に、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておく第4のステップと、

オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブ

ジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、登録されている出力コードの各パターンを使用した回数を測定する第5のステップと、

前記コードパターンを登録する第4のステップによる登録内容および前記コードサイズを測定する第5のステップによる測定結果を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換える第6のステップとを含むことを特徴とする言語処理方法。

【請求項5】 前記コードパターンを登録する第4のステップが、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録するステップを含むことを特徴とする請求項4に記載の言語処理方法。

【請求項6】 前記コードパターン選択ステップが、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定するステップを含むことを特徴とする請求項4または請求項5に記載の言語処理方法。

## 【発明の詳細な説明】

## 【0001】

【発明が属する技術分野】 本発明は、言語処理装置および言語処理方法に関し、特にプログラム作成者が指定したコードサイズの範囲内で実行速度の速いオブジェクトを生成することができる言語処理装置及び言語処理方法に関する。

## 【0002】

【従来の技術】 言語処理プログラムが入力したソースプログラムを翻訳し、その結果として生成したファイルをオブジェクト・モジュール・ファイル（以下、オブジェクトと呼ぶ）と言う。

【0003】 マイクロコンピュータ、特にシングルチップマイクロコンピュータにおいては、コストの低減を図るため、性能面で制限が生じる傾向にある。従って、言語処理プログラムにおいては、言語処理を行った結果であるオブジェクトの実行速度（以下、単に実行速度と呼ぶ）が速いこと、及びオブジェクトのサイズ（以下、コードサイズと呼ぶ）が小さいことが望まれている。

【0004】 従来の言語処理装置では、入力されたソースプログラムに対して、コードサイズと実行速度のどちらかを重視した最適化を行ったオブジェクトの出力を行っている。

【0005】 図10に従来の言語処理装置の構成を示す。

【0006】 図10において、言語処理装置82は、入力したソース・プログラム・ファイル81を構文解析部

83で解析する。構文解析の終了後、コード生成部84に制御を移し、解析された結果およびプログラム作成者の指定に基づいて命令を選択して出力コードを生成する。コード生成の終了後、オブジェクト出力部85に制御を移し、オブジェクト・モジュール・ファイル86を出力する。

【0007】コード生成部84において、言語処理装置82が複数の出力コードパターンを生成し、いずれかの出力コードパターンを選択できる場合には、プログラム作成者の選択により出力コードパターンの選択を行う。すなわち、コードサイズを優先する出力を指定した場合には、コードサイズ優先コード生成部841に制御を移し、コードサイズが小さくなるようにコード生成を行う。コードサイズ優先コード生成部841では、実行速度の向上よりもコードサイズの縮小を優先するため、生成されたオブジェクトの実行速度が遅くなることがあり得る。一方、プログラム作成者が実行速度を優先する出力を指定した場合には、実行速度優先コード生成部842に制御を移し、実行速度が速くなるようにコード生成を行う。実行速度優先コード生成部842では、コードサイズの縮小よりも実行速度の向上を優先するため、コードサイズが増大することがあり得る。

【0008】次に、図11を参照して従来の処理のアルゴリズムを説明する。

【0009】従来の言語処理方式では、まず、プログラム作成者が入力したソースプログラム・ファイルの構文解析を行う(ステップ901)。次に、プログラム作成者がコードサイズを優先する指定をしたか、または実行速度を優先する指定をしたかを判断し(ステップ902)、コードサイズを優先する指定があった場合には、コード生成で複数の出力コードパターンを選択できるところで、コードサイズが小さくなるように出力コードを生成する(ステップ903)。一方、実行速度を優先する指定があった場合には、コード生成で複数の出力コードパターンを選択できるところで、実行速度が速くなるように出力コードを生成する(ステップ904)。そして、出力コードの生成が終了した後、オブジェクトを出力する(ステップ905)。

【0010】このような従来技術の例として、例えば文献「マイクロソフトC オプティマイジング コンパイラ ユーザーズガイド」(1988年マイクロソフト株式会社)に開示された技術がある。同文献には、言語処理装置実行時のオプションにより、オブジェクトのコードサイズを優先する出力を行うか、オブジェクトの実行速度を優先する出力を行うかを選択して出力コードを生成する技術が記載されている。

【0011】

【発明が解決しようとする課題】上述した従来の言語処理技術では、プログラム作成者がオブジェクトに対してコードサイズの縮小と実行速度の向上とを同時に望んだ

場合に、コードサイズと実行速度の両方に対してプログラム作成者が望む条件を十分に満たすオブジェクトを出力することが困難であるという問題点があった。その理由は、上記従来の技術では、コードサイズの縮小を優先するか、実行速度の向上を優先するかの二者択一の選択しかできず、コードサイズを優先する指定では実行速度が遅くなる場合があり、実行速度を優先する指定ではコードサイズが大きくなってしまふ場合があるためである。コードサイズ及び実行速度は命令コードに依存するが、最小かつ最速の命令が常に存在するとは限らない。従って、コードサイズを最小にすると実行速度が遅くなることがあり、実行速度を最速にするとコードサイズが大きくなることがある。

【0012】本発明の目的は、プログラム作成者が指定したコードサイズで、実行速度の速いオブジェクトを生成することができる言語処理装置および言語処理方法を提供することである。

【0013】本発明の他の目的は、プログラム作成者自身がオブジェクトのコードサイズを自由に決定できる言語処理装置および言語処理方法を提供することである。

【0014】

【課題を解決するための手段】上記の目的を達成するため、本発明の言語処理装置は、ソースプログラムを入力して解析する構文解析部と、解析された結果に基づいて命令コードを生成するコード生成部と、生成された命令コードをオブジェクト・モジュール・ファイルに出力するオブジェクト出力部とを備える言語処理装置において、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておくコードパターン登録部と、オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、前記コードパターン登録部に登録されている出力コードの各パターンを使用した回数を測定するコードサイズ測定部と、前記コードサイズ測定部の測定結果と前記コードパターン登録部の登録内容を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部とを備えることを特徴とする。

【0015】請求項2の本発明の言語処理装置は、前記コードパターン登録部が、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録することを特徴と

する。

【0016】請求項3の本発明の言語処理装置は、前記コードパターン選択部が、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定することを特徴とする。

【0017】また、上記の目的を達成する本発明の言語処理方法は、ソースプログラムを入力して解析する第1のステップと、解析された結果に基づいて命令コードを生成する第2のステップと、生成された命令コードをオブジェクト・モジュール・ファイルに出力する第3のステップとを有する言語処理方法において、前記ソースプログラムを解析する第1のステップと命令コードを生成する第2のステップとの間に、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておく第4のステップと、オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、登録されている出力コードの各パターンを使用した回数を測定する第5のステップと、前記コードパターンを登録する第4のステップによる登録内容および前記コードサイズを測定する第5のステップによる測定結果を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換える第6のステップとを含むことを特徴とする。

【0018】請求項5の本発明の言語処理方法は、前記コードパターンを登録する第4のステップが、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録するステップを含むことを特徴とする。

【0019】請求項6の本発明の言語処理方法は、前記コードパターン選択ステップが、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定するステップを含むことを特徴とする。

【0020】

【発明の実施の形態】以下、本発明の実施例について図面を参照して詳細に説明する。

【0021】図1は、本発明の第1の実施例による言語処理装置の構成を示すブロック図である。

【0022】図示のように、言語処理装置12は、ソース・プログラム・ファイル11を入力して内容を解析する構文解析部13と、構文解析の結果を元にコード生成を行うコード生成部17と、コード生成部17で生成さ

れたコードをオブジェクト・モジュール・ファイル19として出力するオブジェクト出力部18を備えると共に、コードサイズを優先する時と実行速度を優先する時とで出力コードが異なるコードパターンの情報が登録してあるコードパターン登録部16と、コードサイズを測定しコードパターン登録部16に登録してあるコードパターンの使用回数を測定するコードサイズ測定部14と、出力コードをプログラム作成者の指定したコードサイズ以内にするため出力コードパターンを選択するコードパターン選択部15とを備える。以上の構成において、構文解析部13とコード生成部17とオブジェクト出力部18の機能は、従来のものと同様である。

【0023】コードパターン登録部16は、コードサイズを優先した場合と実行速度を優先した場合とでコード生成が異なる部分に関して、コードサイズを優先した場合のコードパターンと、実行速度を優先した場合のコードパターンと、コードサイズを優先した場合のコードパターンと実行速度を優先した場合のコードパターンとのコードサイズの差とを登録しておく。コードサイズを優先した場合と実行速度を優先した場合とでオブジェクトのコードサイズが異なる場合は、コードパターン登録部16に登録されたコードパターンが少なくとも1つは存在し、登録されたコードパターンによって生成コードの違いが生じることになる。生成コードの違いが生じると、コードサイズを優先した場合と実行速度を優先した場合とのオブジェクトのコードサイズに差が生じ、一般に、実行速度を優先した場合のコードサイズが大きくなる。

【0024】コードサイズ測定部14は、コードサイズを優先した場合のオブジェクトのコードサイズ、および実行速度を優先した場合のオブジェクトのコードサイズを測定する。コードサイズの測定は、コード生成部17と同様の処理方法で実現可能である。測定した2つのコードサイズ値はコードパターン選択部15の処理において使用される。

【0025】コードサイズ測定部14によるコードサイズ測定処理と並行して、コードパターン登録部16に登録されたコードパターンが、それぞれ何回使用されたかを測定し、コードパターン登録部16に各コードパターンの使用回数を保存する。コードパターン登録部16に保存された各コードパターンの使用回数は、コードパターン選択部15の処理において使用される。

【0026】コードパターン選択部15は、コードパターン登録部16に保存された各コードパターンに対して、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うかを選択するための情報を設定する。設定した情報を参照することによって、オブジェクトのコードサイズがプログラム作成者の指定したコードサイズ以内に収まるように、実行速度を優先した場合のオブジェクトの生成コードの一部または全部

を、コードサイズを優先した場合の生成コードに置き換えることができる。

【0027】コード生成部17は、コードパターン選択部15による選択結果にしたがって、コードサイズ優先コード生成部171または実行速度優先コード生成部172においてコード生成を行う。コードサイズ優先コード生成部171では、コードサイズが小さくなるようにコード生成を行い、実行速度優先コード生成部172では、実行速度が速くなるようにコード生成を行う。コード生成部17でのコード生成は、コードサイズを優先したコード生成と、実行速度を優先したコード生成が混在することになる。コード生成の終了後は、オブジェクト出力部18に制御を移し、オブジェクトを出力する。

【0028】次に、図2及び図3のフローチャートを参照して本実施例の処理の詳細を説明する。

【0029】まず、プログラム作成者が希望するコードサイズを指定する(図2、ステップ201)。指定した値は、メモリ上の特定された領域(「user」と称す)に格納される。

【0030】プログラム作成者が希望するコードサイズを指定すると、構文解析部13が、ソース・プログラム・ファイル11の内容を構文解析する(ステップ202)。構文解析が終了すると、コードサイズ測定部14が、コードサイズを優先した場合のオブジェクトのコードサイズを測定する(ステップ203)。ステップ203のコードサイズ測定処理と並行して、コードパターン登録部16に登録されているコードパターンの使用回数を測定する(ステップ204)。

【0031】図4はコードパターン登録部16の登録内容の例である。コードパターン登録部16には、コードサイズを優先した場合の生成コード1A、2A、・・・、実行速度を優先した場合の生成コード1B、2B、・・・、コードサイズを優先した場合の生成コードと実行速度を優先した場合の生成コードとのコードサイズ差d[1]、d[2]、・・・、及びコードパターンの使用回数cnt[1]、cnt[2]、・・・がそれぞれメモリ上の領域として確保されている。k番目のコードパターンについては、コードサイズを優先した場合の生成コードkA、実行速度を優先した場合の生成コードkB、kAとkBとのコードサイズ差d[k]が登録されている。kAとkBとは生成コードは違うが、同じ動作をするコードである。ステップ204で測定した使用回数は、領域cnt[k]に設定される。

【0032】ステップ203のコードサイズ測定により得られたコードサイズは、メモリ上の領域(「code」と称す)に格納される(ステップ205)。

【0033】ステップ205が終了すると、コードサイズ測定部14が、構文解析部11での構文解析の結果を再度参照し、実行速度を優先した場合のオブジェクトのコードサイズを測定する(ステップ206)。ステップ

206のコードサイズ測定により得られたコードサイズは、メモリ上の領域(「speed」と称す)に格納される(ステップ207)。

【0034】ステップ207の終了後は、コードパターン選択部15に制御を移す。まず、出力予定のオブジェクトのコードサイズの初期設定を行い、メモリ上の領域(「size」と称す)に格納する(ステップ208)。初期設定値は、実行速度を優先した場合のコードサイズ「speed」とする。

【0035】ステップ208の終了後は、コードパターン登録部16に登録されている各コードパターンについて、メモリ上の領域flag[1]、flag[2]、・・・を「0」に初期化する(ステップ209)。flag[1]、flag[2]、・・・とは、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うかを判断するためのフラグを表す。フラグflag[k]は、コードパターン登録部16のk番目に登録されているコードパターンについてのフラグを表す。フラグflag[]は、「0」のときにフラグが降りた状態を示し、「0」以外のときにフラグが立った状態を示すものとする。コード生成では、それぞれのコードパターンに対し、フラグが立っているならばコードサイズを優先するコード生成を行い、フラグが降りているならば実行速度を優先するコード生成を行うことになる。ステップ209のフラグflag[]の初期化後は、フラグflag[]はすべて降りた状態になるので、このままコード生成を行ったと仮定すると、複数のコード生成が可能なすべての記述において、実行速度を優先したコード生成を行うことになる。

【0036】ステップ209のフラグflag[]の初期化が終了すると、プログラム作成者が指定したコードサイズ「user」とコードサイズを優先した場合のコードサイズ「code」とを比較する(図3、ステップ210)。プログラム作成者が指定したコードサイズ「user」が、コードサイズを優先した場合のコードサイズ「code」以下である場合は、コードサイズを優先するコード生成を行うフラグflag[1]、flag[2]、・・・をすべて立て(ステップ211)、コード生成部17に制御を移し、ステップ217以降のコード生成処理を行う。

【0037】ステップ210において、プログラム作成者が指定したコードサイズ「user」が、コードサイズを優先した場合のコードサイズ「code」よりも大きい場合は、コードパターン登録部16に登録されているコードパターンの登録番号を表すメモリ上の領域「i」を「1」に初期化し(ステップ212)、プログラム作成者が指定したコードサイズ「user」と出力予定のオブジェクトのコードサイズ[user]を比較する(ステップ213)。プログラム作成者が指定したコードサイズ「user」、出力予定のオブジェクトの

コードサイズ「user」よりも小さい場合は、フラグflag[i]を立て、コードパターン登録部16に登録されているi番目のコードパターンについて、すべての使用箇所コードサイズ優先のコード生成を行うことにする(ステップ214)。

【0038】ステップ214において、一部のコード生成が実行速度を優先するコード生成からコードサイズを優先するコード生成に代わるにより、出力予定のオブジェクトのコードサイズは、コードパターン登録部16のi番目に登録されているコードサイズ差d[i]と使用回数cnt[i]との積の分だけ小さくなるので、出力予定のオブジェクトのコードサイズ「size」からd[i]とcnt[i]の積を減算した値を、出力予定のオブジェクトのコードサイズ「size」として再設定する(ステップ215)。

【0039】ステップ215における出力予定のオブジェクトのコードサイズ「size」の再設定が終了すると、コードパターン登録部16のコードパターンの登録番号「i」の内容に「1」を加え(ステップ216)、ステップ213に戻って、プログラム作成者が指定したコードサイズ「user」と再設定した出力予定のオブジェクトのコードサイズ「size」とを比較する。

【0040】ステップ213において、プログラム作成者が指定したコードサイズ「user」が、出力予定のオブジェクトのコードサイズ「size」以上である場合には、コード生成部17に制御を移し、ステップ217以降のコード生成処理を行う。

【0041】コード生成部17では、コード生成が終了したか否かを判断する(ステップ217)。コード生成が終了していない場合は、複数の生成コードが存在するか否かを判断する(ステップ218)。これは従来の処理と同じである。生成コードが1つしかない場合は、従来の方法と同様にコード生成を行う(ステップ219)。複数の生成コードが存在する場合は、コードパターン登録部16の何番目に登録されたコードパターンであるかを判断する(ステップ220)。ステップ220で求めたコードパターンの登録番号を「k」とする。

【0042】この後、コードパターン登録部16のk番目に登録されているコードパターンに対して、フラグflag[k]が立っているか否かを判断する(ステップ221)。フラグflag[k]が立っていれば、コードサイズ優先コード生成部17に制御を移し、コードサイズを優先するコードパターンでコード生成を行う(ステップ222)。

また、フラグflag[k]が降りていれば、実行速度優先コード生成部17に制御を移し、実行速度を優先するコードパターンでコード生成を行う(ステップ223)。ステップ219、ステップ222、またはステップ223においてコード生成を行った後は、ステップ217に戻り、コード生成が終了したか否かを判断する。コード生成が終了していなければ

次のコード生成を行う。すべてのコード生成が終了したならば、オブジェクト出力部18に制御を移し、オブジェクト・モジュール・ファイル19を出力する。

【0043】以上の処理を、具体的な例を挙げて説明する。図5は、コードパターン登録部16の登録内容の一例である。図示のように、この例では、コードサイズの単位をbyte、実行速度の単位をclockとし、6パターンのコードパターンが登録されていたものとする。コードパターンの使用回数cnt[1]、cnt[2]、・・・は、ステップ204の処理において設定される。

【0044】所定のソース・プログラム・ファイルについて、コードサイズ測定部14におけるコードサイズの測定により、プログラム作成者が指定したオブジェクトのコードサイズ「user」が1500byte、コードサイズを優先した場合のオブジェクトのコードサイズ「code」が1200byte、実行速度を優先した場合のオブジェクトのコードサイズ「speed」が1600byteであるものとする。また、コードパターン登録部16に登録されているコードパターンの使用回数cnt[i]が、登録番号順に22回、26回、34回、52回、40回、68回であるものとする。

【0045】以上の条件で、ステップ208において、出力予定のオブジェクトのコードサイズ「size」は、実行速度を優先した場合のオブジェクトのコードサイズ「speed」と同じ1600byteに設定される。

【0046】ステップ209において、複数の生成コードが存在する場合にコードサイズを優先するコード生成を行うかどうかを判断するためのフラグflag[i]のすべてを「0」に初期化する。

【0047】ステップ210におけるコードサイズ「user」(1500byte)とコードサイズ「code」(1200byte)との比較で、コードサイズ「user」はコードサイズ「code」より大きいので、ステップ212に処理を移してコードパターン登録部16のコードパターンの登録番号を表すメモリ上の領域「i」を「1」に初期化する。

【0048】ステップ213におけるコードサイズ「user」(1500byte)とコードサイズ「size」(1600byte)との比較において、コードサイズ「user」はコードサイズ「size」より小さいので、ステップ214で、フラグflag[1]を立てる。図5から、コードサイズ差d[1]は1byte、コードパターン使用回数cnt[1]は22回なので、ステップ215において、コードサイズ「size」は(1\*22)byte減って1678byteとなり、ステップ216において、コードパターンの登録番号「i」は「2」となる。

【0049】ステップ216の処理が終了すると、ステ

ップ213に戻って、再びコードサイズ「user」(1500 byte)とコードサイズ「size」(1578 byte)とを比較する。コードサイズ「size」はコードサイズ「user」より大きいので、ステップ214においてフラグflag[2]を立てる。図5から、コードサイズ差d[2]は2 byte、コードパターン使用回数cnt[2]は26回なので、ステップ215において、コードサイズ「size」は(2\*26) byte減って1626 byteとなり、ステップ216において、コードパターンの登録番号「i」は「3」となる。

【0050】ステップ213に戻って、コードサイズ「user」(1500 byte)とコードサイズ「size」(1526 byte)とを比較すると、コードサイズ「size」はコードサイズ「user」より大きいので、ステップ214において、フラグflag[3]を立てる。図5から、コードサイズ差d[3]は1 byte、コードパターン使用回数cnt[3]は34回なので、ステップ215において、コードサイズ「size」は(1\*34) byte減って1492 byteとなり、ステップ216において、コードパターンの登録番号「i」は「4」となる。

【0051】ステップ213に戻って、コードサイズ「user」(1500 byte)とコードサイズ「size」(1492 byte)とを比較すると、コードサイズ「size」はコードサイズ「user」以下となるので、ステップ217以降のコード生成に処理を移す。

【0052】コード生成においては、コードパターン登録部16に登録されているコードパターンについて、登録番号が1~3のコードパターンはフラグflag[]が立っているので、コードサイズを優先するコード生成を行い、登録番号が4~6のコードパターンはフラグflag[]が降りているので、実行速度を優先するコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクトを出力する。

【0053】出力されたオブジェクトは、コードサイズが1492 byteとなり、プログラム作成者の希望するオブジェクトのコードサイズ1500 byte以下という条件を満たしている。また、実行速度を優先するコードパターンが含まれているために、コードサイズを優先した場合のオブジェクトに比して実行速度が速くなる。

【0054】次に、本発明の第2の実施例について図面を参照して詳細に説明する。

【0055】上記の第1の実施例では、コードパターン登録部16に登録されている各コードパターンについて、フラグflag[]が立っているか否かによって、コードサイズを優先するコード生成か実行速度を優先するコード生成かを決定し、決定後はそのコードパターン

を使用する箇所では常に同じコード生成を行っていた。また、第1の実施例で登録されているコードパターンには、登録順番についての規則を設けていないため、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換える順番に関する効率は考慮していない。本実施例では、実行速度の劣化をより抑えたコード生成を行うために、2つの改善を行っている。第1の改善は、登録されている各コードパターンに対して、生成コードの置き換えに関する優先順位を持たせることである。第2の改善は、登録されている各コードパターンに対して、コードサイズを優先するコード生成と実行速度を優先するコード生成とを混在させることによって、出力予定のオブジェクトのサイズをプログラム作成者によって指定されたコードサイズ以内にするための、生成コードの置き換えを最小限に留めることである。

【0056】本実施例の言語処理装置の構成は、コードパターン登録部16に追加登録する項目があること以外は、図1に示した第1の実施例の言語処理装置の構成と同様であるため、全体の構成の説明は省略する。

【0057】図8は、本実施例のコードパターン登録部16の登録内容である。コードパターン登録部16には、図4に示した第1の実施例の登録内容に加え、実行速度を優先した場合の生成コードとコードサイズを優先した場合の生成コードとの実行速度差c[1]、c[2]、・・・がメモリ上の領域として確保され、登録されている。c[k]はコードパターン登録部16のk番目に登録された生成コードkAと生成コードkBとの実行速度差を表す。

【0058】第1の実施例では、コードパターン登録部16に登録されているコードパターンの登録順番には規則がなかったが、本実施例では登録順番に優先順位を付け、優先順位の高いコードパターンから順番に登録しておく。優先順位の高いコードパターンとは、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合に、実行速度の劣化が少なく、かつコードサイズの縮小に効果的なコードパターンである。コードパターン登録部16に登録されたコードサイズ差d[]から実行速度差c[]を減算した値d[]-c[]を優先基準とし、d[]-c[]の値が大きいコードパターンを優先し、値が同じものについては実行速度差c[]が小さいコードパターンを優先する。優先順位を付けることにより、コードパターン選択部15は、実行速度を優先するコード生成からコードサイズを優先するコード生成への置き換えを、実行速度の劣化が少ないコードパターンから順番に行うことが可能となる。

【0059】また、本実施例ではコードパターン選択部15の処理が前記だ1の実施例の場合と異なる。第1の実施例では、フラグflag[1]、flag[2]、・・・によって、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うか



を判断していた。本実施例では、コードパターン登録部16に登録されている各コードパターンに対して、コードサイズを優先するコード生成を行う回数を設定し、設定した回数だけコードサイズを優先するコード生成を行い、残りの回数は実行速度を優先するコード生成を行う。

【0060】以下、図6及び図7のフローチャートを参照して本実施例の処理の詳細を説明する。

【0061】まず、図2に図示された、コードパターン選択部16によるステップ208までの処理を行う。処理内容は第1の実施例と同様であるため説明を省略する。

【0062】ステップ208までの処理が終了すると、コードパターン登録部16に登録されている各コードパターンについて、コードサイズを優先するコードパターンの使用回数を表すメモリ上の領域  $use[1]$ 、 $use[2]$ 、... を「0」に初期化する（図7、ステップ501）。 $use[k]$  は、コードパターン登録部16のk番目に登録されているコードパターンについて、コードサイズを優先するコードパターンの使用回数を表す。

【0063】ステップ501の初期化が終了すると、プログラム作成者が指定したコードサイズ「user」とコードサイズを優先した場合のコードサイズ「code」とを比較する（ステップ502）。プログラム作成者が指定したコードサイズ「user」が、コードサイズを優先した場合のコードサイズ「code」以下である場合は、コードサイズを優先するコードパターンの使用回数  $use[i]$  に、コードパターンの使用回数  $cnt[i]$  の値を設定する（ステップ503）。コードパターン登録部16のk番目に登録されているコードパターンに関しては、コードサイズを優先するコードパターンの使用回数  $use[k]$  に、コードパターンの使用回数  $cnt[k]$  の値が設定される。ステップ503の設定により、すべてのコードパターンにおいてコードサイズを優先するコード生成を行うことになる。ステップ503の処理が終了すると、コード生成部17に制御を移し、ステップ511以降のコード生成処理を行う。

【0064】ステップ502において、プログラム作成者が指定したコードサイズ「user」が、コードサイズを優先した場合のコードサイズ「code」よりも大きい場合は、コードパターン登録部16に登録されているコードパターンの登録番号を表すメモリ上の領域「i」を「1」に初期化し（ステップ504）、プログラム作成者が指定したコードサイズ「user」と出力予定のオブジェクトのコードサイズ「size」を比較する（ステップ505）。プログラム作成者が指定したコードサイズ「user」が、出力予定のオブジェクトのコードサイズ「size」以上である場合は、コード生成部17に制御を移し、ステップ511以降のコード

生成処理を行う。

【0065】ステップ505において、プログラム作成者が指定したコードサイズ「user」が、出力予定のオブジェクトのコードサイズ「size」より小さい場合は、出力予定のオブジェクトのコードサイズ「size」から、コードパターン登録部16のi番目に登録されているコードサイズ差  $d[i]$  と使用回数  $cnt$

「i」との積を減算した値と、プログラム作成者が指定したオブジェクトのコードサイズ「user」とを比較する（ステップ506）。ステップ506において比較に用いられる値  $size - d[i] * cnt[i]$  は、コードパターン登録部16のi番目のコードパターンを使用しているすべての箇所で、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合の出力予定のオブジェクトのコードサイズである。

【0066】ステップ506において、プログラム作成者が指定したコードサイズ「user」が、 $size - d[i] * cnt[i]$  以下である場合は、コードパターン登録部16のi番目に登録されているコードパターンに関して、コードサイズを優先するコードパターンの使用回数  $use[i]$  にコードパターンの使用回数  $cnt[i]$  の値を設定し（ステップ507）、出力予定のオブジェクトのコードサイズ「size」に、 $size - d[i] * cnt[i]$  を再設定する（ステップ508）。ステップ508における出力予定のオブジェクトのコードサイズ「size」の再設定が終了すると、コードパターン登録部16のコードパターンの登録番号「i」の内容に「1」加え（ステップ509）、ステップ505に戻って、プログラム作成者が指定したコードサイズ「user」と再設定した出力予定のオブジェクトのコードサイズ「size」とを比較する。

【0067】ステップ506において、プログラム作成者が指定したコードサイズ「user」が、 $size - d[i] * cnt[i]$  より大きい場合は、コードパターン登録部16のi番目に登録されているコードパターンに関して、出力予定のオブジェクトのコードサイズ「user」以下になるような、コードサイズを優先するコードパターンの使用回数を計算し、 $use[i]$  に設定する（ステップ510）。ステップ510で  $use[i]$  に設定する値は、 $(user - size) / d[i]$  以上の最小の整数値である。ステップ510における  $use[i]$  の設定が終了すると、コード生成部17に制御を移し、ステップ511以降のコード生成処理を行う。

【0068】コード生成部17では、コード生成が終了したか否かを判断する（図7、ステップ511）。コード生成が終了していない場合は、複数の生成コードが存在するか否かを判断する（ステップ512）。これは従来の処理と同じである。生成コードが1つしかない

場合は、従来の方法と同様にコード生成を行う（ステップ513）。複数の生成コードが存在する場合は、コードパターン登録部16の何番目に登録されたコードパターンであるかを判断する（ステップ514）。ステップ514で求めたコードパターンの登録番号をkとする。

【0069】コードパターン登録部16のk番目に登録されているコードパターンに対して、コードサイズを優先するコードパターンの使用回数use[k]が「0」か否かを調べ（ステップ515）、use[k]が「0」であるならば、実行速度優先コード生成部172に制御を移し、実行速度を優先するコードパターンでコード生成を行う（ステップ516）。use[k]が「0」でなければ、コードサイズ優先コード生成部171に制御を移し、コードサイズを優先するコードパターンでコード生成を行い（ステップ517）、use[k]の内容から「1」を引く（ステップ518）。ステップ513、ステップ516、またはステップ517、518においてコード生成を行った後は、ステップ511に戻り、コード生成が終了したか否かを判断し、終了していなければ次のコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクト出力部18に制御を移し、オブジェクト・モジュール・ファイル19を出力する。

【0070】以上の処理を、具体的な例を挙げて説明する。図9は、コードパターン登録部16の登録内容の一例である。図示のように、この例では、コードサイズの単位をbyte、実行速度の単位をclockとする。図9に登録されているコードパターンは、図5と同じものであるが、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合に、実行速度の劣化が少なく、かつコードサイズの縮小に効果的なコードパターンから順に登録されている。優先基準としては、コードサイズ差d[]から実行速度差c[]を引いた値が大きいコードパターンを優先し、値が同じものについては、実行速度差c[]が小さいコードパターンを優先する。コードパターンの使用回数cnt[1]、cnt[2]、・・・については、ステップ204において設定される。

【0071】所定のソース・プログラム・ファイルについて、コードサイズ測定部14におけるコードサイズの測定により、プログラム作成者が指定したオブジェクトのコードサイズ「user」が1500byte、コードサイズを優先した場合のオブジェクトのコードサイズ「code」が1200byte、実行速度を優先した場合のオブジェクトのコードサイズ「speed」が1600byteであるものとする。また、コードパターン登録部16に登録されているコードパターンの使用回数cnt[]が、登録番号順に52回、68回、40回、22回、26回、34回であるものとする。

【0072】ステップ208において、出力予定のオブ

ジェクトのコードサイズ「size」は、実行速度を優先した場合のオブジェクトのコードサイズspeedと同じ1600byteに設定される。

【0073】ステップ501において、コードサイズを優先するコードパターンの使用回数use[]のすべてを「0」に初期化する。

【0074】ステップ502におけるコードサイズ「user」（1500byte）とコードサイズ「code」（1200byte）との比較で、コードサイズ「user」はコードサイズ「code」より大きいので、ステップ504に処理を移してコードパターン登録部16のコードパターンの登録番号を表すメモリ上の領域「i」を「1」に初期化する。

【0075】ステップ505におけるコードサイズ「user」（1500byte）とコードサイズ「size」（1600byte）との比較で、コードサイズ「user」はコードサイズ「size」より小さいので、ステップ506に処理を移す。図9から、コードサイズ差d[1]は2byte、コードパターン使用回数cnt[1]は52回なので、ステップ506において、コードサイズuser（1500byte）とコードサイズsize-d[1]\*cnt[1]（（1600-2\*52）byte）を比較する。

【0076】コードサイズ「user」はコードサイズsize-d[1]\*cnt[1]より大きいので、ステップ510に処理を移して、use[1]に（user-size）/d[1]以上の最小の整数値を設定する。（1600-1500）/2=50であることから、use[1]には50を設定する。即ち、1番目に登録されているコードパターンの使用回数52回のうち、50回はコードサイズを優先するコード生成を行い、残りの2回は実行速度を優先するコード生成を行うことになる。

【0077】1番目のコードパターンについてのコード生成を行う場合は、use[1]の値が「0」になるまで、ステップ517に処理を移して、コードサイズを優先するコード生成を行い、ステップ518においてuse[1]の内容から「1」を引く。use[1]の値が「0」になったならば、ステップ516に処理を移して、実行速度を優先するコード生成を行う。2番目以降のコードパターンについては、use[]の値が「0」であるので、ステップ516に処理を移して、実行速度を優先するコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクトを出力する。

【0078】出力されたオブジェクトはコードサイズが1500byteとなり、プログラム作成者の希望するオブジェクトのコードサイズ1500byte以下という条件を満たしている。また、実行速度を優先したコードパターンが含まれているために、コードサイズを優先した場合のオブジェクトよりも実行速度が速くなる。さ

らに、実行速度の劣化を抑えた生成コードの置き換えを行っているので、第 1 の実施例よりも実行速度が速くなっている。

【0079】以上のように、本発明では、コードサイズを優先した場合のコードサイズおよび登録したコードパターンの使用回数、実行速度を優先した場合のコードサイズを測定し、実行速度を優先した場合の生成コードの一部または全部を、プログラム作成者が指定したコードサイズ以内に収まるまで、コードサイズを優先した場合の生成コードに置き換えてオブジェクトを出力する。このため、従来のコードサイズを優先した場合の生成コードと、実行速度を優先した場合の生成コードとの間の任意のサイズのコードを生成することができ、結果として、プログラム作成者が指定したコードサイズ以内で、実行速度の速いオブジェクトを出力することができる。

【0080】以上好ましい実施例をあげて本発明を説明したが、本発明は必ずしも上記実施例に限定されるものではない。

#### 【0081】

【発明の効果】以上説明したように、本発明の言語処理装置及び言語処理方法によれば、実行速度を優先するコード生成の一部を、コードサイズを優先するコード生成に置き換えることにより、実行速度の劣化を抑えると共にオブジェクトのコードサイズの調整ができるため、オブジェクトの実行速度をあまり損なわずに、コードサイズの短縮ができるという効果がある。

【0082】これにより、実行速度を優先するコード生成を行ったオブジェクトが、プログラム作成者の希望するコードサイズに収まらなかった場合にプログラム作成者自身がオブジェクトを書き換えてオブジェクトのコードサイズを調整するといったプログラム作成者の手間を省くことができるという効果がある。

【0083】また、オブジェクトのコードサイズを調節して、実行速度を向上することができるため、コードサイズを優先するコード生成を行ったオブジェクトが、プログラム作成者の希望するコードサイズよりずっと小さくなった場合に、オブジェクトのコードサイズをプログラム作成者の希望するコードサイズまで増やすことによって、より実行速度の速いオブジェクトを作成すること

ができるという効果がある。

#### 【図面の簡単な説明】

【図 1】 本発明の第 1 の実施例による言語処理装置の構成を示すブロック図である。

【図 2】 第 1 の実施例によるコードサイズ測定部からコード生成部までの動作を示すフローチャートである。

【図 3】 第 1 の実施例によるコードサイズ測定部からコード生成部までの動作を示すフローチャートである。

【図 4】 第 1 の実施例による言語処理装置のコードパターン登録部の登録内容を示す図である。

【図 5】 第 1 の実施例による言語処理装置のコードパターン登録部の登録内容の具体例を示す図である。

【図 6】 本発明の第 2 の実施例によるコードパターン選択部からコード生成部までの動作を示すフローチャートである。

【図 7】 本発明の第 2 の実施例によるコードパターン選択部からコード生成部までの動作を示すフローチャートである。

【図 8】 第 2 の実施例による言語処理装置のコードパターン登録部の登録内容を示す図である。

【図 9】 第 2 の実施例による言語処理装置のコードパターン登録部の登録内容の具体例を示す図である。

【図 10】 従来の言語処理装置の構成を示すブロック図である。

【図 11】 従来の言語処理装置の動作を示すフローチャートである。

#### 【符号の説明】

- 1 1 ソースプログラムファイル
- 1 2 言語処理装置
- 1 3 構文解析部
- 1 4 コードサイズ測定部
- 1 5 コードパターン選択部
- 1 6 コードパターン登録部
- 1 7 コード生成部
- 1 8 オブジェクト出力部
- 1 9 オブジェクトモジュールファイル
- 1 7 1 コードサイズ優先コード生成部
- 1 7 2 実行速度優先コード生成部

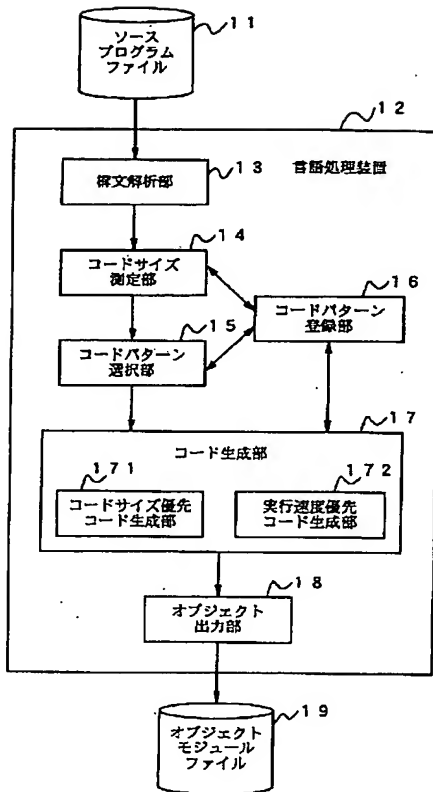
【図 4】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ登録	パターン使用回数
1	1 A	1 B	d [1]	cnt [1]
2	2 A	2 B	d [2]	cnt [2]
⋮	⋮	⋮	⋮	⋮
n	n A	n B	d [n]	cnt [n]

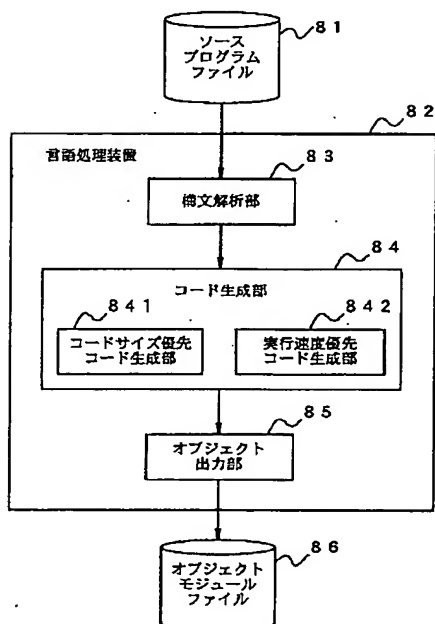
【図 5】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	パターン使用回数
1	1 A (2byte, 6clock)	1 B (8byte, 4clock)	1 byte	2 2
2	2 A (4byte, 9clock)	2 B (6byte, 6clock)	2 byte	2 6
3	3 A (3byte, 9clock)	3 B (4byte, 5clock)	1 byte	3 4
4	4 A (5byte, 8clock)	4 B (7byte, 7clock)	2 byte	5 2
5	5 A (2byte, 8clock)	5 B (5byte, 5clock)	3 byte	4 0
6	6 A (3byte, 7clock)	6 B (4byte, 6clock)	1 byte	6 8

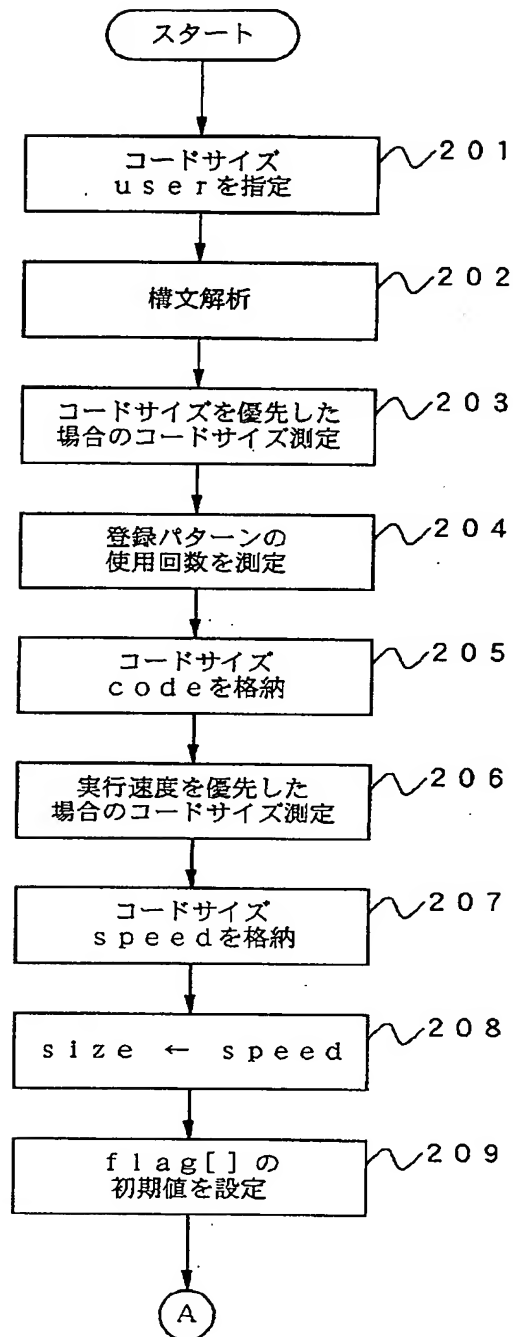
【図1】



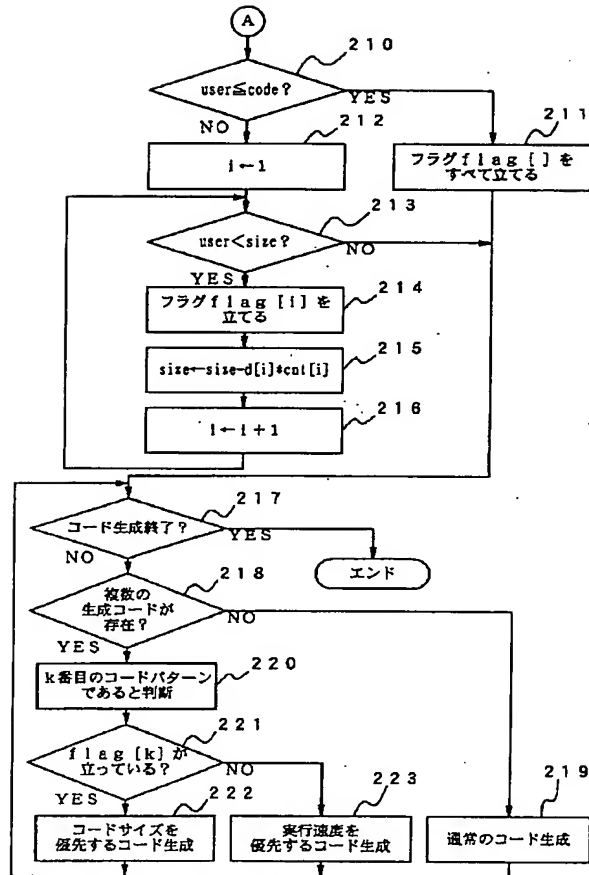
【図10】



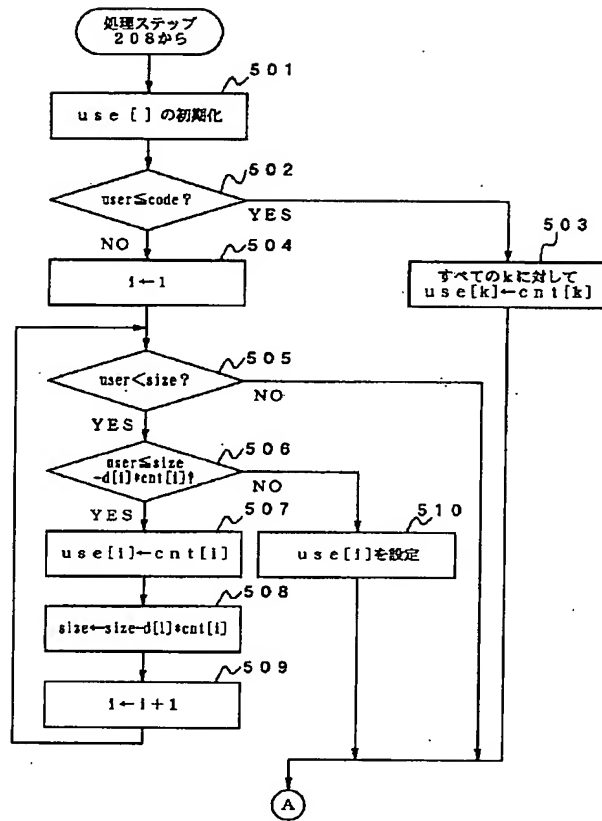
【図2】



【図3】



【図6】



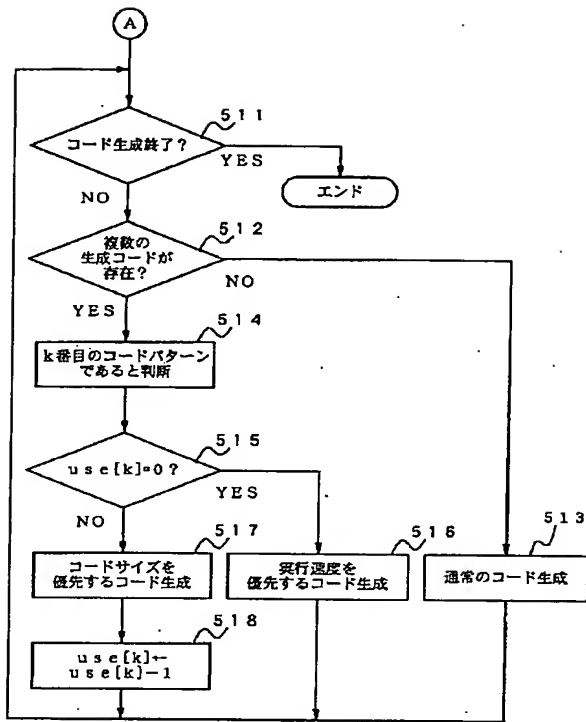
【図9】

【図8】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	実行速度差	パターン使用回数
1	1 A	1 B	d[1]	c[1]	cnt[1]
2	2 A	2 B	d[2]	c[2]	cnt[2]
⋮	⋮	⋮	⋮	⋮	⋮
n	n A	n B	d[n]	c[n]	cnt[n]

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	実行速度差	パターン使用回数
1	1 A (5byte, 8clock)	1 B (7byte, 7clock)	2 byte	1 clock	5 2
2	2 A (3byte, 7clock)	2 B (4byte, 6clock)	1 byte	1 clock	6 8
3	3 A (2byte, 8clock)	3 B (5byte, 5clock)	3 byte	3 clock	4 0
4	4 A (2byte, 6clock)	4 B (3byte, 4clock)	1 byte	2 clock	2 2
5	5 A (4byte, 9clock)	5 B (6byte, 6clock)	2 byte	3 clock	2 6
6	6 A (3byte, 8clock)	6 B (4byte, 5clock)	1 byte	4 clock	3 4

【図7】



【図11】

